Board 348: Poster - Unified Regular Expression Antipattern Language (UREAL)

Joseph Roy Teahen, Michigan Technological University
Daniel Masker, Michigan Technological University
Dr. Leo C. Ureel II, Michigan Technological University

Leo C. Ureel II is an Assistant Professor in Computer Science and in Cognitive and Learning Sciences at Michigan Technological University. He has worked extensively in the field of educational software development. His research interests include intelligent learning environments, computer science education, and Artificial Intelligence

Dr. Laura E Brown, Michigan Technological University
Dr. Michelle E Jarvie-Eggart P.E., Michigan Technological University

Dr. Jarvie-Eggart is a registered professional engineer with over a decade of experience as an environmental engineer. She is an Assistant Professor of Engineering Fundamentals at Michigan Technological University. Her research interests include technology adoption, problem based and service learning, and sustainability.

Dr. Jon Sticklen, Michigan Technological University

Jon Sticklen is an Associate Professor with the Engineering Fundamentals Department (EF) and Affiliated Faculty with the Department of Cognitive and Learning Sciences (CLS). He served as Chair of EF from 2014-2020, leading a successful effort to design a

Poster - Unified Regular Expression Antipattern Language

Abstract

In this work in progress poster, we discuss the unification of regular expressions to find antipatterns in WebTA. Unified Regular Expression Antipattern Language (UREAL) seeks to unify regular expression (regex) antipatterns where the only difference is syntax. UREAL captures syntactic differences by language through regex tokenization. Instead of specifying the specific regex for each code structure, we specify a UREAL token which is usable across languages. We then use these UREAL tokens to create the regex antipatterns. We are able to automatically substitute language-specific regex into UREAL expressions when using them to parse a given language to find antipatterns. By unifying the regex in this way, we are able to reduce development overhead for new languages, increasing the time that can be spent encoding new antipatterns and providing quality feedback. Increasing the effectiveness and language diversity of WebTA will help students improve their programming skills regardless of chosen language and will help instructors draw upon a deeper antipattern library. This design-based research will be evaluated on understandability, portability, and antipattern coverage.

Introduction

WebTA is a multi-language code critic designed to detect, report, and explain novice antipatterns to beginner programmers across many engineering and computing disciplines [1–4]. Novice antipatterns are mistakes made in code that seem correct, but contain logical and structural fallacies. WebTA finds these antipatterns, displays them to the student, and offers immediate and meaningful, novice-targeted feedback to fix the problem. WebTA currently supports Java, MATLAB, and Python, with more languages in development [5].

Problem

Many of the antipatterns in WebTA are specified using regular expressions. Writing this regex can be difficult, as consideration needs to be made for both structure of the code and nuances the language ignores, such as whitespace and newlines. Similar antipatterns appear across the different languages, with subtle differences based on the language's representation of logical structures such as if, while, and or operator statements. While these structures are syntactically different, they are semantically identical [6]. For each language we add to WebTA, many antipatterns need to be rewritten due to these syntactical differences. This increases development overhead and lessens the effectiveness of new languages due to a lesser corpus of antipatterns. Additionally, the nature of regular expressions makes them hard to understand when first looking at them. This increases development time for even the most experienced of regex users. UREAL seeks to address the portability of antipatterns to new languages, increase the understanding of each antipattern, and lessen development times.

Design

UREAL is broken down into multiple sections defined by cross-language specification

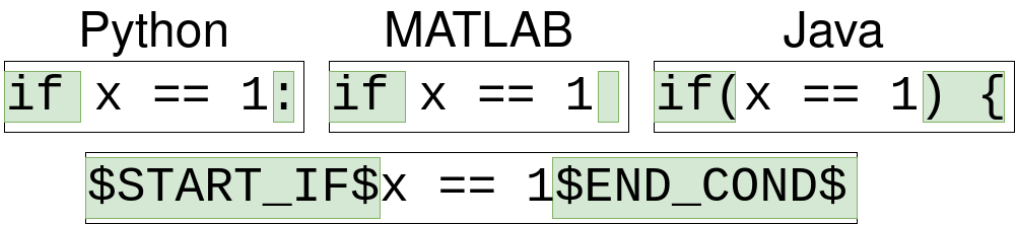

Figure 1: Example UREAL Expression

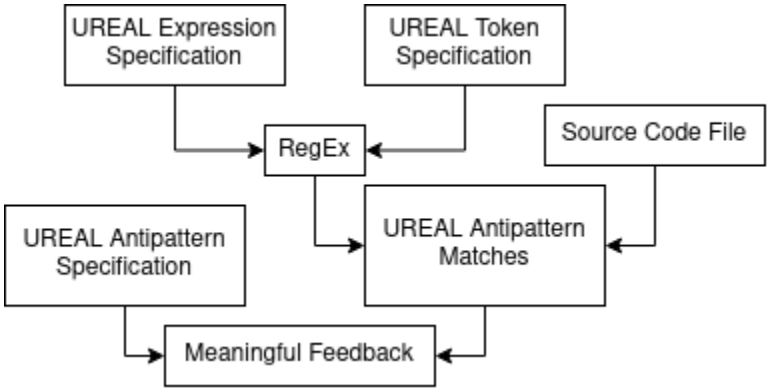

Figure 2: UREAL Design Flow

changes. Our goal is to move from the specific regex representations of language syntax to a generalized antipattern representation. We first move from the specific syntax of individual languages by specifying UREAL tokens. UREAL tokens match single token names to multiple regular expressions defined for each language. Once a UREAL token is defined for a language, that regex no longer needs to be specified. UREAL tokens are specified in a single file for each language. We can then specify language-agnostic expressions using UREAL tokens combined with standard regex. We call these UREAL expressions. In a UREAL expression, tokens are expanded into their language-specific regular expressions on demand by each code critic. In effect, if the "shape" of a piece of source code in two different languages is similar, we are able to write one UREAL expression to match it. An example of this is found in Figure 1. In this example, two UREAL tokens, `$START_IF$` and `$END_COND$` are used to refer to the beginning and end of an if condition. These tokens are expanded to match the specific syntax of the target language. Expanded UREAL expressions are used identically to standard regex.

Using UREAL expressions, we create UREAL antipatterns. These consist of a descriptive name, the UREAL expression specifying the antipattern, and novice-geared feedback covering why the antipattern is incorrect and some guidance on how to fix it. It may optionally contain information specific to the language the antipattern is found in. The novice-geared feedback is generated by content experts and modified by instructors for the given course. This allows for feedback to be in the voice of the instructor and at the level of the student.

This design flow is summarized in Figure 2. The mappings of UREAL Tokens are specified

for a given language, then they are combined with a general UREAL expression to create regex to parse that language. The regex is then matched against the student source code file to find instances of UREAL antipatterns. Finally, these instances are matched to their descriptions and return meaningful feedback to the student.

Future Work

The design stage is complete and current work is moving into enactment and analysis [7]. In the preliminary work for this poster, we have created several proof-of-concept programs. Enactment will consist of integrating UREAL as a Java package into WebTA, modifying the antipattern database, and constructing the language tokenizations. A subset of language structures will be constructed for initial evaluation (i.e., if, while, etc.). This analysis will be done on three metrics: ease of understanding and construction, portability, and total coverage. Ease of understanding and construction will evaluate how intuitive UREAL is to use. Both speed of usage and reduction in mistakes will be analyzed. Portability will be evaluated by how many UREAL antipatterns, once specified, have zero false positives or negatives across languages. Total coverage will be evaluated per language by the reduction of language-specific regex use.

Acknowledgements

This work was funded by the National Science Foundation award #2142309. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] L. C. Ureel II and C. Wallace, "WebTA: Automated iterative critique of student programming assignments," in *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, IEEE, 2015.

[2] L. C. Ureel and C. R. Wallace, "WebTA: Online Code Critique and Assignment Feedback," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 1111–1111, 2018.

[3] L. C. Ureel II, *Critiquing Antipatterns In Novice Code*. PhD thesis, Michigan Technological University, Houghton, MI, Aug 2020.

[4] L. C. Ureel II, L. E. Brown, J. Sticklen, M. Jarvie-Eggart, and M. Benjamin, "Work in Progress: The RICA Project: Rich, Immediate Critique of Antipatterns in Student Code," in *Educational Data Mining in Computer Science Education (CSEDM) Workshop*, July 2022.

[5] L. C. Ureel II, "Integrating a Colony of Code Critiquers into WebTA," in *Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice"*, 2021.

[6] J. Teahen, D. T. Masker, L. C. Ureel, M. Eggart, J. Sticklen, and L. E. Brown, "Extending the Usability of WebTA with Unified ASTs and Errors," in *2023 IEEE*

Frontiers in Education Conference (FIE), (Los Alamitos, CA, USA), pp. 1–5, IEEE Computer Society, oct 2023.

[7] The Design-Based Research Collective, "Design-based research: An emerging paradigm for educational inquiry," *Educational Researcher*, vol. 32, no. 1, pp. 5–8, 2003.